Description Logics

Franz Baader[1], Ian Horrocks[2], and Ulrike Sattler[2]

[1] Institut für Theoretische Informatik, TU Dresden, Germany
baader@tcs.inf.tu-dresden.de
[2] Department of Computer Science, University of Manchester, UK
{horrocks,sattler}@cs.man.ac.uk

Summary. In this chapter, we explain what description logics are and why they make good ontology languages. In particular, we introduce the description logic $\mathcal{SHIQ}$, which has formed the basis of several well-known ontology languages, including OWL. We argue that, without the last decade of basic research in description logics, this family of knowledge representation languages could not have played such an important rôle in this context.

Description logic reasoning can be used both during the design phase, in order to improve the quality of ontologies, and in the deployment phase, in order to exploit the rich structure of ontologies and ontology based information. We discuss the extensions to $\mathcal{SHIQ}$ that are required for languages such as OWL and, finally, we sketch how novel reasoning services can support building DL knowledge bases.

1 Introduction

The aim of this section is to give a brief introduction to description logics, and to argue why they are well-suited as ontology languages. In the remainder of the chapter we will put some flesh on this skeleton by providing more technical details with respect to the theory of description logics, and their relationship to state of the art ontology languages. More detail on these and other matters related to description logics can be found in [6].

Ontologies

There have been many attempts to define what constitutes an ontology, perhaps the best known (at least amongst computer scientists) being due to Gruber: "an ontology is an explicit specification of a conceptualisation" [47].[3] In this context, a conceptualisation means an abstract model of some aspect of the world, taking the form of a definition of the properties of important

[3] This was later elaborated to "a formal specification of a shared conceptualisation" [21].

concepts and relationships. An explicit specification means that the model should be specified in some unambiguous language, making it amenable to processing by machines as well as by humans.

Ontologies are becoming of increasing importance in fields such as knowledge management, information integration, cooperative information systems, information retrieval and electronic commerce. One application area which has recently seen an explosion of interest is the so called *Semantic Web* [18], where ontologies are set to play a key rôle in establishing a common terminology between agents, thus ensuring that different agents have a shared understanding of terms used in semantic markup.

The effective use of ontologies requires not only a well-designed and well-defined ontology language, but also support from reasoning tools. Reasoning is important both to ensure the quality of an ontology, and in order to exploit the rich structure of ontologies and ontology based information. It can be employed in different phases of the ontology life cycle. During ontology design, it can be used to test whether concepts are non-contradictory and to derive implied relations. In particular, one usually wants to compute the concept hierarchy, i.e., the partial ordering of named concepts based on the subsumption relationship. Information on which concept is a specialization of another, and which concepts are synonyms, can be used in the design phase to test whether the concept definitions in the ontology have the intended consequences or not. This information is also very useful when the ontology is deployed.

Since it is not reasonable to assume that all applications will use the same ontology, interoperability and integration of different ontologies is also an important issue. Integration can, for example, be supported as follows: after the knowledge engineer has asserted some inter-ontology relationships, the integrated concept hierarchy is computed and the concepts are checked for consistency. Inconsistent concepts as well as unintended or missing subsumption relationships are thus signs of incorrect or incomplete inter-ontology assertions, which can then be corrected or completed by the knowledge engineer.

Finally, reasoning may also be used when the ontology is deployed. As well as using the pre-computed concept hierarchy, one could, for example, use the ontology to determine the consistency of facts stated in annotations, or infer relationships between annotation instances and ontology classes. More precisely, when searching web pages annotated with terms from the ontology, it may be useful to consider not only exact matches, but also matches with respect to more general or more specific terms—where the latter choice depends on the context. However, in the deployment phase, the requirements on the efficiency of reasoning are much more stringent than in the design and integration phases.

Before arguing why description logics are good candidates for such an ontology language, we provide a brief introduction to and history of description logics.

Description Logics

Description logics (DLs) [6, 16, 30] are a family of knowledge representation languages that can be used to represent the knowledge of an application domain in a structured and formally well-understood way. The name *description logics* is motivated by the fact that, on the one hand, the important notions of the domain are described by concept *descriptions*, i.e., expressions that are built from atomic concepts (unary predicates) and atomic roles (binary predicates) using the concept and role constructors provided by the particular DL. On the other hand, DLs differ from their predecessors, such as semantic networks and frames, in that they are equipped with a formal, *logic*-based semantics.

In this introduction, we only illustrate some typical constructors by an example. Formal definitions are given in Section 2. Assume that we want to define the concept of "A man that is married to a doctor and has at least five children, all of whom are professors." This concept can be described with the following concept description:

$$\mathsf{Human} \sqcap \neg\mathsf{Female} \sqcap \exists\mathsf{married}.\mathsf{Doctor} \sqcap (\geq 5\,\mathsf{hasChild}) \sqcap \forall\mathsf{hasChild}.\mathsf{Professor}$$

This description employs the Boolean constructors *conjunction* ($\sqcap$), which is interpreted as set intersection, and *negation* ($\neg$), which is interpreted as set complement, as well as the *existential restriction* constructor ($\exists R.C$), the *value restriction* constructor ($\forall R.C$), and the *number restriction* constructor ($\geq n\,R$). An individual, say Bob, belongs to $\exists\mathsf{married}.\mathsf{Doctor}$ if there exists an individual that is married to Bob (i.e., is related to Bob via the $\mathsf{married}$ role) and is a doctor (i.e., belongs to the concept Doctor). Similarly, Bob belongs to $(\geq 5\,\mathsf{hasChild})$ iff he has at least five children, and he belongs to $\forall\mathsf{hasChild}.\mathsf{Professor}$ iff all his children (i.e., all individuals related to Bob via the $\mathsf{hasChild}$ role) are professors.

In addition to this description formalism, DLs are usually equipped with a terminological and an assertional formalism. In its simplest form, *terminological axioms* can be used to introduce names (abbreviations) for complex descriptions. For example, we could introduce the abbreviation $\mathsf{HappyMan}$ for the concept description from above. More expressive terminological formalisms allow the statement of constraints such as

$$\exists\mathsf{hasChild}.\mathsf{Human} \sqsubseteq \mathsf{Human},$$

which says that only humans can have human children. A set of terminological axioms is called a TBox. The *assertional formalism* can be used to state properties of individuals. For example, the assertions

$$\mathsf{HappyMan}(\mathsf{BOB}), \quad \mathsf{hasChild}(\mathsf{BOB}, \mathsf{MARY})$$

state that Bob belongs to the concept $\mathsf{HappyMan}$ and that Mary is one of his children. A set of such assertions is called an ABox, and the named individuals that occur in ABox assertions are called ABox individuals.

Description logic systems provide their users with various inference capabilities that deduce implicit knowledge from the explicitly represented knowledge. The *subsumption* algorithm determines subconcept-superconcept relationships: C is subsumed by D iff all instances of C are necessarily instances of D, i.e., the first description is always interpreted as a subset of the second description. For example, given the definition of HappyMan from above, HappyMan is subsumed by $\exists$hasChild.Professor—since instances of HappyMan have at least five children, all of whom are professors, they also have a child that is a professor. The *instance* algorithm determines instance relationships: the individual i is an instance of the concept description C iff i is always interpreted as an element of C. For example, given the assertions from above and the definition of HappyMan, MARY is an instance of Professor. The *consistency* algorithm determines whether a knowledge base (consisting of a set of assertions and a set of terminological axioms) is non-contradictory. For example, if we add $\neg$Professor(MARY) to the two assertions from above, then the knowledge base containing these assertions together with the definition of HappyMan from above is inconsistent.

In order to ensure a reasonable and predictable behavior of a DL system, these inference problems should at least be decidable for the DL employed by the system, and preferably of low complexity. Consequently, the expressive power of the DL in question must be restricted in an appropriate way. If the imposed restrictions are too severe, however, then the important notions of the application domain can no longer be expressed. Investigating this trade-off between the expressivity of DLs and the complexity of their inference problems has been one of the most important issues in DL research. Roughly, the research related to this issue can be classified into the following four phases.

Phase 1 (1980–1990) was mainly concerned with implementation of systems, such as KLONE, K-REP, BACK, and LOOM [24, 69, 78, 68]. These systems employed so-called *structural subsumption algorithms*, which first normalize the concept descriptions, and then recursively compare the syntactic structure of the normalized descriptions [71]. These algorithms are usually relatively efficient (polynomial), but they have the disadvantage that they are complete only for very inexpressive DLs, i.e., for more expressive DLs they cannot detect all the existing subsumption/instance relationships. At the end of this phase, early formal investigations into the complexity of reasoning in DLs showed that most DLs do not have polynomial-time inference problems [23, 72]. As a reaction, the implementors of the CLASSIC system (the first industrial-strength DL system) carefully restricted the expressive power of their DL [77, 22].

Phase 2 (1990–1995) started with the introduction of a new algorithmic paradigm into DLs, so-called *tableau-based algorithms* [85, 39, 52]. They work on propositionally closed DLs (i.e., DLs with full Boolean operators) and are complete also for expressive DLs. To decide the consistency of a knowledge base, a tableau-based algorithm tries to construct a model of it by break-

ing down the concepts in the knowledge base, thus inferring new constraints on the elements of this model. The algorithm either stops because all attempts to build a model failed with obvious contradictions, or it stops with a "canonical" model. Since in propositionally closed DLs, subsumption and satisfiability can be reduced to consistency, a consistency algorithm can solve all inference problems mentioned above. The first systems employing such algorithms (KRIS and CRACK) demonstrated that optimized implementations of these algorithm lead to an acceptable behavior of the system, even though the worst-case complexity of the corresponding reasoning problems is no longer in polynomial time [9, 27]. This phase also saw a thorough analysis of the complexity of reasoning in various DLs [39, 40, 38]. Another important observation was that DLs are very closely related to modal logics [83].

Phase 3 (1995–2000) is characterized by the development of inference procedures for very expressive DLs, either based on the tableau-approach [56, 58] or on a translation into modal logics [35, 36, 34, 37]. Highly optimized systems (FaCT, RACE, and DLP [53, 48, 76]) showed that tableau-based algorithms for expressive DLs lead to a good practical behavior of the system even on (some) large knowledge bases. In this phase, the relationship to modal logics [35, 84] and to decidable fragments of first-order logic was also studied in more detail [19, 74, 45, 43, 44], and applications in databases (like schema reasoning, query optimization, and integration of databases) were investigated [28, 29, 31].

We are now at the beginning of *Phase 4*, where industrial strength DL systems employing very expressive DLs and tableau-based algorithms are being developed, with applications like the Semantic Web or knowledge representation and integration in bio-informatics in mind.

Description Logics as Ontology Languages

As already mentioned above, high quality ontologies are crucial for many applications, and their construction, integration, and evolution greatly depends on the availability of a well-defined semantics and powerful reasoning tools. Since DLs provide for both, they should be ideal candidates for ontology languages. That much was already clear ten years ago, but at that time there was a fundamental mismatch between the expressive power and the efficiency of reasoning that DL systems provided, and the expressivity and the large knowledge bases that ontologists needed [41]. Through the basic research in DLs of the last 10–15 years that we have summarized above, this gap between the needs of ontologist and the systems that DL researchers provide has finally become narrow enough to build stable bridges.

The suitability of DLs as ontology languages has been highlighted by their role as the foundation for several web ontology languages, including OWL, an ontology language standard developed by the W3C Web-Ontology Working

Group.[4] OWL has a syntax based on RDF Schema, but the basis for its design is the expressive DL $\mathcal{SHIQ}$ [59],[5] and the developers have tried to find a good compromise between expressiveness and the complexity of reasoning. Although reasoning in $\mathcal{SHIQ}$ is decidable, it has a rather high worst-case complexity (ExpTime). Nevertheless, highly optimized $\mathcal{SHIQ}$ reasoners such as FaCT++ [92], Racer [50] and Pellet [88] behave quite well in practice.

Let us point out some of the features of $\mathcal{SHIQ}$ that make this DL expressive enough to be used as an ontology language. Firstly, $\mathcal{SHIQ}$ provides number restrictions that are more expressive than the ones introduced above (and employed be earlier DL systems). With the *qualified number restrictions* available in $\mathcal{SHIQ}$, as well as being able to say that a person has at most two children (without mentioning the properties of these children):

$$(\leq 2\,\mathsf{hasChild}),$$

one can also specify that there is at most one son and at most one daughter:

$$(\leq 1\,\mathsf{hasChild}.\neg\mathsf{Female}) \sqcap (\leq 1\,\mathsf{hasChild}.\mathsf{Female}).$$

Secondly, $\mathcal{SHIQ}$ allows the formulation of complex terminological axioms like "humans have human parents":

$$\mathsf{Human} \sqsubseteq \exists \mathsf{hasParent}.\mathsf{Human}.$$

Thirdly, $\mathcal{SHIQ}$ also allows for *inverse roles*, *transitive roles*, and *subroles*. For example, in addition to hasChild one can also use its inverse hasParent, one can specify that hasAncestor is transitive, and that hasParent is a subrole of hasAncestor.

It has been argued in the DL and the ontology community that these features play a central role when describing properties of aggregated objects and when building ontologies [81, 90, 42]. The actual use of a DL providing these features as the underlying logical formalism of the web ontology language OWL [55] substantiates this claim [90].[6]

Finally, we would like to mention briefly three extensions to $\mathcal{SHIQ}$ that are often used in ontology languages (we will discuss them in more detail in Section 5).

Complex roles are often required in ontologies. For example, when describing complex physically composed structures it may be desirable to express the fact that damage to a part of the structure implies damage to the structure as a whole. This feature is particularly important in medical ontologies: it is supported in the Grail DL [79], which was specifically designed for use with medical terminology, and in another medical terminology application using

[4] `http://www.w3.org/2001/sw/WebOnt/`

[5] To be exact, it is based on $\mathcal{SHOIN}$.

[6] OWL does not, however, support the use of the more expressive qualified number restrictions.

the comparatively inexpressive DL $\mathcal{ALC}$, a rather complex "work around" is performed in order to capture this kind of information [86].[7]

It is quite straightforward to extend $\mathcal{SHIQ}$ so that this kind of propagation can be expressed: simply allow for the use of complex roles in role inclusion axioms. E.g., $\mathsf{hasLocation} \circ \mathsf{partOf} \sqsubseteq \mathsf{hasLocation}$ expresses the fact that things located in part of something are also located in the thing as a whole. Although this leads to undecidability in general, syntactic restrictions can be devised that lead to a decidable logic [57].

Concrete domains [7, 67] integrate DLs with concrete sets such as the real numbers, integers, or strings, and built-in predicates such as comparisons $\leq$, comparisons with constants ≤ 17, or $\mathsf{isPrefixOf}$. This supports the modelling of concrete properties of abstract objects such as the age, the weight, or the name of a person, and the comparison of these concrete properties. Unfortunately, in their unrestricted form, concrete domains can have dramatic effects on the decidability and computational complexity of the underlying DL [67].

Nominals are special concept names that are to be interpreted as singleton sets. Using a nominal Turing, we can describe all those computer scientists that have met Turing by $\mathsf{CSientist} \sqcap \exists \mathsf{hasMet}.\mathsf{Turing}$. Again, nominals can have dramatic effects on the complexity of a logic [91].

2 The Expressive Description Logic $\mathcal{SHIQ}$

In this section, we present syntax and semantics of the expressive DL $\mathcal{SHIQ}$ (although, as we will see in Section 4, the DL underlying OWL is, in some respects, slightly more expressive). Moreover, we will concentrate on the *terminological* formalism, i.e., the part that supports the definition of the relevant concepts in an application domain, and the statement of constraints that restrict interpretations to the intended ones. The *assertional* formalism will not be introduced here due to space limitations and since it only plays a minor role in ontology engineering. The interested reader is referred to [6, 82] for assertional DL formalisms in general, and to [49, 60] for assertional reasoning for $\mathcal{SHIQ}$.

In contrast to most of the DLs considered in the literature, which concentrate on constructors for defining concepts, the DL $\mathcal{SHIQ}$ [58] also allows for rather expressive roles. Of course, these roles can then be used in the definition of concepts. We start with the definition of $\mathcal{SHIQ}$-roles, and then continue with the definition of $\mathcal{SHIQ}$-concepts.

Definition 1 (Syntax and semantics of $\mathcal{SHIQ}$-roles). *Let* $\mathbf{R}$ *be a set of* role names*, which is partitioned into a set* $\mathbf{R}_+$ *of transitive roles and a set* $\mathbf{R}_\mathsf{P}$ *of normal roles. The set of all* $\mathcal{SHIQ}$-roles *is* $\mathbf{R} \cup \{r^- \mid r \in \mathbf{R}\}$*, where* r^- *is*

[7] In this approach, so-called *SEP-triplets* are used both to compensate for the absence of transitive roles in $\mathcal{ALC}$, and to express the propagation of properties across a distinguished "part-of" role.

called the inverse *of the role* r. *A* role inclusion axiom *is of the form* $r \sqsubseteq s$, *where* r, s *are* $\mathcal{SHIQ}$*-roles. A* role hierarchy *is a finite set of role inclusion axioms.*

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *consists of a set* $\Delta^{\mathcal{I}}$, *called the* domain *of* $\mathcal{I}$, *and a function* $\cdot^{\mathcal{I}}$ *that maps every role to a subset of* $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ *such that, for all* $p \in \mathbf{R}$ *and* $r \in \mathbf{R}_+$,

$$\langle x, y\rangle \in p^{\mathcal{I}} \quad \textit{iff} \quad \langle y, x\rangle \in (p^-)^{\mathcal{I}},$$
$$\textit{if } \langle x, y\rangle \in r^{\mathcal{I}} \textit{ and } \langle y, z\rangle \in r^{\mathcal{I}} \textit{ then } \langle x, z\rangle \in r^{\mathcal{I}}.$$

An interpretation $\mathcal{I}$ satisfies *a role hierarchy* $\mathcal{R}$ *iff* $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ *for each* $r \sqsubseteq s \in \mathcal{R}$; *such an interpretation is called a* model *of* $\mathcal{R}$.

The unrestricted use of these roles in all of the concept constructors of $\mathcal{SHIQ}$ (to be defined below) would lead to an undecidable DL [58]. Therefore, we must first define an appropriate subset of all $\mathcal{SHIQ}$-roles. This requires some more notation.

1. The inverse relation on binary relations is symmetric, i.e., the inverse of r^- is again r. To avoid writing role expressions such as r^{--}, r^{---}, etc., we define a function Inv, which returns the inverse of a role:
$$\mathsf{Inv}(r) := \begin{cases} r^- & \text{if } r \text{ is a role name,} \\ s & \text{if } r = s^- \text{ for a role name } s. \end{cases}$$
2. Since set inclusion is transitive and an inclusion relation between two roles transfers to their inverses, a given role hierarchy $\mathcal{R}$ implies additional inclusion relationships. To account for this fact, we define $\sqsubseteq^*_{\mathcal{R}}$ as the reflexive-transitive closure of
$$\sqsubseteq_{\mathcal{R}} := \mathcal{R} \cup \{\mathsf{Inv}(r) \sqsubseteq \mathsf{Inv}(s) \mid r \sqsubseteq s \in \mathcal{R}\}.$$
We use $r \equiv_{\mathcal{R}} s$ as an abbreviation for $r \sqsubseteq^*_{\mathcal{R}} s$ and $s \sqsubseteq^*_{\mathcal{R}} r$. In this case, every model of $\mathcal{R}$ interprets these roles as the same binary relation.
3. Obviously, a binary relation is transitive iff its inverse is transitive. Thus, if $r \equiv_{\mathcal{R}} s$ and r or $\mathsf{Inv}(r)$ is transitive, then any model of $\mathcal{R}$ interprets s as a transitive binary relation. To account for such implied transitive roles, we define the following function Trans:
$$\mathsf{Trans}(s, \mathcal{R}) := \begin{cases} \text{true} & \text{if } r \in \mathbf{R}_+ \text{ or } \mathsf{Inv}(r) \in \mathbf{R}_+ \text{ for some } r \text{ with } r \equiv_{\mathcal{R}} s \\ \text{false} & \text{otherwise.} \end{cases}$$
4. A role r is called *simple* w.r.t. $\mathcal{R}$ iff $\mathsf{Trans}(s, \mathcal{R}) = \text{false}$ for all $s \sqsubseteq^*_{\mathcal{R}} r$.

Definition 2 (Syntax and semantics of $\mathcal{SHIQ}$-concepts). *Let* N_C *be a set of* concept names. *The set of* $\mathcal{SHIQ}$-concepts *is the smallest set such that*

1. *every concept name $A \in N_C$ is a $\mathcal{SHIQ}$-concept,*
2. *if C and D are $\mathcal{SHIQ}$-concepts and r is a $\mathcal{SHIQ}$-role, then $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall r.C$, and $\exists r.C$ are $\mathcal{SHIQ}$-concepts,*
3. *if C is a $\mathcal{SHIQ}$-concept, r is a* simple *$\mathcal{SHIQ}$-role, and $n \in \mathbb{N}$, then $(\leqslant n\ r.C)$ and $(\geqslant n\ r.C)$ are $\mathcal{SHIQ}$-concepts.*

The interpretation function $\cdot^{\mathcal{I}}$ of an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ maps, additionally, every concept to a subset of $\Delta^{\mathcal{I}}$ such that

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}, \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}, \qquad \neg C^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}},$$

$$(\exists r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \textit{There is some } y \in \Delta^{\mathcal{I}} \textit{ with } \langle x, y\rangle \in r^{\mathcal{I}} \textit{ and } y \in C^{\mathcal{I}}\},$$

$$(\forall r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \textit{For all } y \in \Delta^{\mathcal{I}}, \textit{ if } \langle x, y\rangle \in r^{\mathcal{I}}, \textit{then } y \in C^{\mathcal{I}}\},$$

$$(\leqslant n\ r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \sharp r^{\mathcal{I}}(x, C) \leqslant n\},$$

$$(\geqslant n\ r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \sharp r^{\mathcal{I}}(x, C) \geqslant n\},$$

where $\sharp M$ denotes the cardinality of the set M, and $r^{\mathcal{I}}(x, C) := \{y \mid \langle x, y\rangle \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$. If $x \in C^{\mathcal{I}}$, then we say that x is an instance *of C in $\mathcal{I}$, and if $\langle x, y\rangle \in r^{\mathcal{I}}$, then y is called an* r-successor *of x in $\mathcal{I}$.*

Concepts can be used to describe the relevant notions of an application domain. The terminology (TBox) introduces abbreviations (names) for complex concepts. In $\mathcal{SHIQ}$, the TBox also allows one to state more complex constraints.

Definition 3. *A* general concept inclusion *(GCI) is of the form $C \sqsubseteq D$, where C, D are $\mathcal{SHIQ}$-concepts. A finite set of GCIs is called a* TBox. *An interpretation $\mathcal{I}$ is a* model *of a TBox $\mathcal{T}$ iff it satisfies all GCIs in $\mathcal{T}$, i.e., $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for each $C \sqsubseteq D \in \mathcal{T}$.*

A concept definition is of the form $A \equiv C$, where A is a concept name. It can be seen as an abbreviation for the two GCIs $A \sqsubseteq C$ and $C \sqsubseteq A$.

Inference problems are defined w.r.t. a TBox and a role hierarchy.

Definition 4. *The concept C is called* satisfiable with respect to the role hierarchy $\mathcal{R}$ and the TBox $\mathcal{T}$ *iff there is a model $\mathcal{I}$ of $\mathcal{R}$ and $\mathcal{T}$ with $C^{\mathcal{I}} \neq \emptyset$. Such an interpretation is called a* model of C *w.r.t. $\mathcal{R}$ and $\mathcal{T}$. The concept D* subsumes *the concept C w.r.t. $\langle \mathcal{R}, \mathcal{T}\rangle$ (written $C \sqsubseteq_{\langle \mathcal{R}, \mathcal{T}\rangle} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{R}$ and $\mathcal{T}$. Two concepts C, D are* equivalent *w.r.t. $\mathcal{R}$ (written $C \equiv_{\langle \mathcal{R}, \mathcal{T}\rangle} D$) iff they subsume each other.*

By definition, equivalence can be reduced to subsumption. In addition, subsumption can be reduced to satisfiability since $C \sqsubseteq_{\langle \mathcal{R}, \mathcal{T}\rangle} D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{R}$ and $\mathcal{T}$.

As mentioned above, most DLs are (decidable) fragments of (first-order) predicate logic [19, 5]. Viewing role names as binary relations and concept

names as unary relations, for example, the role inclusion axiom $r \sqsubseteq s^-$ translates into $\forall x \forall y.r(x,y) \Rightarrow s(y,x)$, and the GCI $A \sqcap \exists r.C \sqsubseteq D \sqcup \forall s^-.E$ translates into

$$\forall x.(A(x) \wedge \exists y.r(x,y) \wedge C(y)) \Rightarrow (D(x) \vee \forall y.s(y,x) \Rightarrow E(x)).$$

This translation preserves the semantics: we can easily view DL interpretations as predicate logic interpretations, and then prove, e.g., that each model of a concept C w.r.t. a TBox $\mathcal{T}$ and a role hierarchy $\mathcal{R}$ is a model of the translation of C conjoined with the (universally quantified) translations of $\mathcal{T}$ and $\mathcal{R}$.

3 Describing Ontologies in $\mathcal{SHIQ}$

In general, an ontology can be formalised in a TBox as follows. Firstly, we restrict the possible worlds by introducing restrictions on the allowed interpretations. For example, to express that, in our world, we want to consider humans, which are either muggles or sorcerers, we can use the GCIs

$$\mathsf{Human} \sqsubseteq \mathsf{Muggle} \sqcup \mathsf{Sorcerer} \quad \text{and} \quad \mathsf{Muggle} \sqsubseteq \neg\mathsf{Sorcerer}.$$

Next, to express that humans have exactly two parents and that all parents and children of humans are human, we can use the following GCI:

$$\mathsf{Human} \sqsubseteq \forall \mathsf{hasParent}.\mathsf{Human} \sqcap (\leqslant 2\ \mathsf{hasParent}.\top) \sqcap (\geqslant 2\ \mathsf{hasParent}.\top) \sqcap \forall \mathsf{hasParent}^-.\mathsf{Human},$$

where $\top$ is an abbreviation for the top concept $A \sqcup \neg A$.[8]

In addition, we consider the *transitive* role $\mathsf{hasAncestor}$, and the role inclusion

$$\mathsf{hasParent} \sqsubseteq \mathsf{hasAncestor}.$$

The next GCI expresses that humans having an ancestor that is a sorcerer are themselves sorcerers:

$$\mathsf{Human} \sqcap \exists \mathsf{hasAncestor}.\mathsf{Sorcerer} \sqsubseteq \mathsf{Sorcerer}.$$

Secondly, we can define the relevant notions of our application domain using concept definitions. Recall that the concept definition $A \equiv C$ stands for the two GCIs $A \sqsubseteq C$ and $C \sqsubseteq A$. A concept name is called *defined* if it occurs on the left-hand side of a definition, and *primitive* otherwise.

We want our concept definitions to have definitional impact, i.e., the interpretation of the primitive concept and role names should uniquely determine the interpretation of the defined concept names. For this, the set of concept definitions together with the additional GCIs must satisfy three conditions:

[8] When the qualifying concept is $\top$, this is equivalent to an unqualified restriction, and it will often be written as such, e.g., $(\leqslant 2\ \mathsf{hasParent})$.

1. There are no multiple definitions, i.e., each defined concept name must occur at most once as a left-hand side of a concept definition.
2. There are no cyclic definitions, i.e., no cyclic dependencies between the defined names in the set of concept definitions.[9]
3. The defined names do not occur in any of the additional GCIs.

In contrast to concept definitions, the GCIs in $\mathcal{SHIQ}$ may well have cyclic dependencies between concept names. An example are the above GCIs describing humans.

As a simple example of a set of concept definitions satisfying the restrictions from above, we define the concepts grandparent and parent:[10]

$$\mathsf{Parent} \equiv \mathsf{Human} \sqcap \exists \mathsf{hasParent}^{-}.\top,$$
$$\mathsf{Grandparent} \equiv \exists \mathsf{hasParent}^{-}.\mathsf{Parent}.$$

The TBox consisting of the above concept definitions and GCIs, together with the fact that hasAncestor is a transitive superrole of hasParent, implies the following subsumption relationship:

$$\mathsf{Grandparent} \sqcap \mathsf{Sorcerer} \sqsubseteq \exists \mathsf{hasParent}^{-}.\exists \mathsf{hasParent}^{-}.\mathsf{Sorcerer},$$

i.e., grandparents that are sorcerers have a grandchild that is a sorcerer. Though this conclusion may sound reasonable given the assumptions, it requires quite some reasoning to obtain it. In particular, one must use the fact that hasAncestor (and thus also $\mathsf{hasAncestor}^{-}$) is transitive, that $\mathsf{hasParent}^{-}$ is the inverse of hasParent, and that we have a GCI that says that children of humans are again humans.

To sum up, a $\mathcal{SHIQ}$-TBox can, on the one hand, axiomatize the basic notions of an application domain (the primitive concepts) by GCIs, transitivity statements, and role inclusions, in the sense that these statements restrict the possible interpretations of the basic notions. On the other hand, more complex notions (the defined concepts) can be introduced by concept definitions. Given an interpretation of the basic notions, the concept definitions uniquely determine the interpretation of the defined notions.

The *taxonomy* of such a TBox is then given by the subsumption hierarchy of the defined concepts. It can be computed using a subsumption algorithm for $\mathcal{SHIQ}$ (see Chapters 23 and 24). The knowledge engineer can test whether the TBox captures her intuition by checking the satisfiability of the defined concepts (since it does not make sense to give a complex definition for the empty concept), and by checking whether their place in the taxonomy corresponds to their intuitive place. The expressive power of $\mathcal{SHIQ}$ together with the fact that one can "verify" the TBox in the sense mentioned above is the main reason for $\mathcal{SHIQ}$ being well-suited as an ontology language [81, 42, 90].

[9] In order to give cyclic definitions definitional impact, one would need to use fixpoint semantics for them [73, 1].

[10] In addition to the role hasParent, which relates children to their parents, we use the concept Parent, which describes all humans having children.

4 $\mathcal{SHIQ}$ and OWL

As already discussed, OWL is a semantic web ontology language whose the semantics can be defined via a translation into an expressive DL.[11] This is not a coincidence—it was a design goal. The mapping allows OWL to exploit formal results from DL research (e.g., regarding the decidability and complexity of key inference problems) and use implemented DL reasoners (e.g., FaCT++ [92], RACER [50] and Pellet [88]) in order to provide reasoning services for OWL applications.

An OWL (Lite or DL) ontology can be seen to correspond to a DL TBox together with a role hierarchy, describing the domain in terms of *classes* (corresponding to concepts) and *properties* (corresponding to roles). An ontology consists of a set of *axioms* that assert, e.g., subsumption relationships between classes or properties.

As in a standard DL, OWL classes may be names or expressions built up from simpler classes and properties using a variety of constructors. The set of constructors supported by OWL, along with the equivalent DL abstract syntax, is summarised in Figure 1.[12] The full XML serialisation of the RDF syntax is not shown as it is rather verbose, e.g., $\mathsf{Human} \sqcap \mathsf{Male}$ would be written as

```
<owl:Class>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Human"/>
    <owl:Class rdf:about="#Male"/>
  </owl:intersectionOf>
</owl:Class>
```

while $(\geqslant 2\ \mathsf{hasChild}.\mathsf{Thing})$ would be written as

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasChild"/>
    <owl:minCardinality
      rdf:datatype="&xsd;NonNegativeInteger">2
  </owl:minCardinality>
</owl:Restriction>
```

Prefixes such as `owl:` and `&xsd;` specify XML namespaces for resources, while `rdf:parseType="Collection"` is an extension to RDF that provides a "shorthand" notation for lisp style lists defined using triples with the properties first and rest (it can be eliminated, but with a consequent increase in verbosity). E.g., the first example above consists of the

[11] In fact there are 3 "species" of OWL: OWL Lite, OWL DL and OWL full, only the first two of which have DL based semantics. The semantics of OWL full is given by an extension to RDF model theory [51].

[12] In fact, there are a few additional constructors provided as "syntactic sugar", but all are trivially reducible to the ones described in Figure 1.

triples $\langle \mathsf{r}_1, \mathsf{owl:intersectionOf}, \mathsf{r}_2 \rangle$, $\langle \mathsf{r}_2, \mathsf{owl:first}, \mathsf{Human} \rangle$, $\langle \mathsf{r}_2, \mathsf{rdfs:type}, \mathsf{Class} \rangle$, $\langle \mathsf{r}_2, \mathsf{owl:rest}, \mathsf{r}_3 \rangle$, etc., where r_i is an anonymous resource, Human stands for a URI naming the resource "Human", and owl : intersectionOf, owl : first, owl : rest and rdfs : type stand for URIs naming the properties in question.

Constructor	DL Syntax	Example
`intersectionOf`	$C_1 \sqcap \ldots \sqcap C_n$	$\mathsf{Human} \sqcap \mathsf{Male}$
`unionOf`	$C_1 \sqcup \ldots \sqcup C_n$	$\mathsf{Doctor} \sqcup \mathsf{Lawyer}$
`complementOf`	$\neg C$	$\neg\mathsf{Male}$
`oneOf`	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	$\{\mathsf{john}\} \sqcup \{\mathsf{mary}\}$
`allValuesFrom`	$\forall P.C$	$\forall\mathsf{hasChild}.\mathsf{Doctor}$
`someValuesFrom`	$\exists r.C$	$\exists\mathsf{hasChild}.\mathsf{Lawyer}$
`hasValue`	$\exists r.\{x\}$	$\exists\mathsf{citizenOf}.\{\mathsf{USA}\}$
`minCardinality`	$(\geqslant n\ r)$	$(\geqslant 2\ \mathsf{hasChild})$
`maxCardinality`	$(\leqslant n\ r)$	$(\leqslant 1\ \mathsf{hasChild})$
`inverseOf`	r^-	$\mathsf{hasChild}^-$

Fig. 1. OWL constructors

As already mentioned, an OWL ontology consists of a set of axioms. Figure 2 summarises the axioms supported by OWL. These axioms make it possible to assert subsumption or equivalence with respect to classes or properties, the disjointness of classes, and the equivalence or non-equivalence of individuals (resources). Moreover, OWL also allows properties of properties (i.e., DL roles) to be asserted. In particular, it is possible to assert that a property is transitive, functional, inverse functional or symmetric.

Axiom	DL Syntax	Example
`subClassOf`	$C_1 \sqsubseteq C_2$	$\mathsf{Human} \sqsubseteq \mathsf{Animal} \sqcap \mathsf{Biped}$
`equivalentClass`	$C_1 \equiv C_2$	$\mathsf{Man} \equiv \mathsf{Human} \sqcap \mathsf{Male}$
`subPropertyOf`	$P_1 \sqsubseteq P_2$	$\mathsf{hasDaughter} \sqsubseteq \mathsf{hasChild}$
`equivalentProperty`	$P_1 \equiv P_2$	$\mathsf{cost} \equiv \mathsf{price}$
`disjointWith`	$C_1 \sqsubseteq \neg C_2$	$\mathsf{Male} \sqsubseteq \neg\mathsf{Female}$
`sameAs`	$\{x_1\} \equiv \{x_2\}$	$\{\mathsf{President\_Bush}\} \equiv \{\mathsf{G\_W\_Bush}\}$
`differentFrom`	$\{x_1\} \sqsubseteq \neg\{x_2\}$	$\{\mathsf{john}\} \sqsubseteq \neg\{\mathsf{peter}\}$
`TransitiveProperty`	$P \in \mathbf{R}_+$	$\mathsf{hasAncestor}^+ \in \mathbf{R}_+$
`FunctionalProperty`	$\top \sqsubseteq (\leqslant 1\ P)$	$\top \sqsubseteq (\leqslant 1\ \mathsf{hasMother})$
`InverseFunctionalProperty`	$\top \sqsubseteq (\leqslant 1\ P^-)$	$\top \sqsubseteq (\leqslant 1\ \mathsf{isMotherOf}^-)$
`SymmetricProperty`	$P \equiv P^-$	$\mathsf{isSiblingOf} \equiv \mathsf{isSiblingOf}^-$

Fig. 2. OWL axioms

This shows that, except for individuals and datatypes, the constructors and axioms of OWL can be translated into $\mathcal{SHIQ}$. In fact, OWL is equivalent to

the extension of $\mathcal{SHIN}$ ($\mathcal{SHIQ}$ with only the $\top$ concept being allowed in qualified number restrictions) with nominals and a simple form of concrete domains (this extension will be discussed in more detail in Section 5).

As discussed in Section 1, establishing a link between ontology languages and DLs allows us to support ontology engineering by providing a range of reasoning services. Such services can be implemented using various algorithmic techniques, including tableaux-based techniques (see Chapter 23) and resolution-based techniques (see Chapter 24).

5 Extensions and variants of $\mathcal{SHIQ}$

As mentioned above, the ontology language OWL extends $\mathcal{SHIQ}$ with nominals and concrete datatypes. In this section, we discuss the consequences of these extensions on the reasoning problems in $\mathcal{SHIQ}$.

Concrete datatypes, as available in OWL, are a very restricted form of concrete domains [7]. For example, using the concrete domain of all nonnegative integers equipped with the $<$ predicate, a (functional) role age relating (abstract) individuals to their (concrete) age, and a (functional) subrole father of $\mathsf{hasParent}$, the following axiom states that children are younger than their fathers:

$$\mathsf{Animal} \sqsubseteq (\mathsf{age} < (\mathsf{father} \circ \mathsf{age})).$$

Extending expressive DLs with concrete domains may easily lead to undecidability [8, 66]. In OWL, however, no datatype predicates are supported—only XML schema datatypes (such as integer and string) and enumerations (such as $\{1, 2, 5, 7\}$) can be used in descriptions. These restrictions are enough to ensure that decidability is not compromised (in fact in [75], decidability of $\mathcal{SHIQ}$ extended with a more general type of concrete domains is shown).

Concerning nominals, things become a bit more complicated. Firstly, we believe that we can use the same (relativised axiomatization) technique as used for $\mathcal{SHIQ}$ in [91] to translate $\mathcal{SHIQ}$ extended with nominals into a fragment of C2, the two-variable fragment of first order logic with counting quantifiers [46, 74]. Since this translation is polynomial, satisfiability and subsumption are decidable in NExpTime. This is optimal since the problem is also NExpTime-hard [91]. Roughly speaking, the combination of GCIs (or transitive roles and role hierarchies), inverse roles, and number restrictions with nominals is responsible for this leap in complexity (from ExpTime for $\mathcal{SHIQ}$ to NExpTime). Until recently, no "practicable" decision procedure for $\mathcal{SHIQ}$ with nominals had been described, where by "practicable" we mean a decision procedure that works in some "goal-directed" way, in contrast to "blindly" guessing a model $\mathcal{I}$ of at most exponential size and then checking whether $\mathcal{I}$ is indeed a model of the input. An extension of the tableaux algorithm for $\mathcal{SHIQ}$ has, however, now been developed [57], has been successfully implemented in the FaCT++ and Pellet systems, and seems to work well on realistic ontologies [87].

Finally, as mentioned above, it is quite straightforward to extend $\mathcal{SHIQ}$, or even $\mathcal{SHOIQ}$, with complex role inclusion axioms. The resulting DL, $\mathcal{SROIQ}$ [54], is the basis for a recent proposal to extend the OWL language, the extended language being called OWL 1.1.[13] In addition to complex role inclusion axioms, OWL 1.1 also supports qualified number restrictions, and more expressive datatypes than OWL.

6 Novel Reasoning Services

As argued in the introduction, standard DL reasoning services (such as satisfiability and subsumption algorithms) can be used in different phases of the ontology life cycle. In the design phase, they can test whether concepts are non-contradictory and can derive implied relations between concepts. However, for these services to be applied, one already needs a sufficiently developed TBox. The result of reasoning can then be used to develop the TBox further. Until now, however, DL systems provide no reasoning support for writing this initial TBox. The development of so-called non-standard inferences in DLs (like computing least common subsumers [32, 13, 63, 65], most specific concepts [10, 64], rewriting [14], approximation [26], and matching [20, 12, 11, 3]) tries to overcome this deficit. In the following, we will sketch how these novel inferences can support building a DL knowledge base.

Assume that the knowledge engineer wants to introduce the definition of a new concept into the TBox. In many cases, she will not develop this new definition from scratch, but rather try to re-use things that are already present in some knowledge base (either the one she is currently building or a previous one). In a chemical process engineering application [80, 70], we have observed two ways in which this is realized in practise:

1. The knowledge engineer decides on the basic structure of the newly defined concept, and then tries to find already defined concepts that have a similar structure. These concepts can then be modified to obtain the new concept.
2. Instead of directly defining the new concept, the knowledge engineer first gives examples of objects that belong to the concept to be defined, and then tries to generalize these examples into a concept definition.

Both approaches can be supported by the non-standard inferences mentioned above, though this kind of support is not yet provided by any of the existing DL systems.

The first approach can be supported by matching concept patterns against concept descriptions. A *concept pattern* is a concept description that may contain variables that stand for descriptions. A *matcher* σ of a pattern D onto the description C replaces the variables by concept descriptions such that the resulting concept $\sigma(D)$ is equivalent to C. For example, assume that the

[13] `http://www.w3.org/Submission/owl11-overview/`

knowledge engineer is looking for concepts concerned with individuals having a son and a daughter sharing some characteristic. This can be expressed by the pattern

$$\exists \mathsf{hasChild}.(\mathsf{Male} \sqcap X) \sqcap \exists \mathsf{hasChild}.(\mathsf{Female} \sqcap X).$$

The substitution $\sigma = \{X \mapsto \mathsf{Tall}\}$ shows that this pattern matches the description $\exists \mathsf{hasChild}.(\mathsf{Male} \sqcap \mathsf{Tall}) \sqcap \exists \mathsf{hasChild}.(\mathsf{Female} \sqcap \mathsf{Tall})$. Note, however, that in some cases the existence of a matcher is not so obvious.

The second approach can be supported by algorithms that compute most specific concepts and least common subsumers. Assume that the examples are given as ABox individuals $i_1, \ldots, i_k$. In a first step, these individuals are generalized into concepts by respectively computing the most specific (w.r.t. subsumption) concepts $C_1, \ldots, C_k$ in the available DL that have these individuals as instances. In a second step, these concepts are generalized into one concept by computing the least common subsumer of $C_1, \ldots, C_k$, i.e., the least concept description (in the available DL) that subsumes $C_1, \ldots, C_k$. In this context, rewriting of concepts comes into play since the concept descriptions produced by the algorithms for computing least common subsumers may be rather large (and thus not easy to comprehend and modify for the knowledge engineer). Rewriting minimizes the size of these description without changing their meaning by introducing names defined in the TBox.

Until now, the results on such non-standard inferences are restricted to DLs that are considerably less expressive than $\mathcal{SHIQ}$. For some of them, they only make sense if used for inexpressive DLs. For example, in DLs that contain the disjunction constructor, the least common subsumer of $C_1, \ldots, C_k$ is simply their disjunction, and computing this is of no help to the knowledge engineer. What one would like to obtain as a result of the least common subsumer computation are the structural similarities between the input concepts.

Thus, support by non-standard inferences can only be given if one uses DLs of restricted expressive power. However, this also makes sense in the context of ontology engineering. In fact, the users that will require the most support are the naive ones, and it is reasonable to assume that they will not use (or even be offered) the full expressive power of the underlying DL. This two-level approach is already present in tools like Protégé [62] , which offer a frame-like user interface. Using this simple interface, one gets only a fragment of the expressive power of OWL. To use the full expressive power, one must type in DL expressions.

Another way to overcome the gap between DLs of different expressive power is to use the approximation inference [26]. Here, one tries to approximate a given concept description C in an expressive DL $\mathcal{L}_1$ by a description D in a less expressive DL $\mathcal{L}_2$. When approximating from above, D should be the least description in $\mathcal{L}_2$ subsuming C, and when approximating from below, D should be the greatest description $\mathcal{L}_2$ subsumed by C.

7 Conclusion

The emphasis in DL research on a formal, logic-based semantics and a thorough investigation of the basic reasoning problems, together with the availability of highly optimized systems for very expressive DLs, makes this family of knowledge representation formalisms an ideal starting point for defining ontology languages. The reasoning services required to support the construction, integration, and evolution of high quality ontologies are provided by state-of-the-art DL systems for very expressive languages.

To be used in practice, these languages will, however, also need DL-based tools that further support knowledge acquisition (i.e., building ontologies), maintenance (i.e., evolution of ontologies), and integration and inter-operation of ontologies. First steps in this direction have already been taken. For example, Protégé [62] and SWOOP [61] are tools that support the development of OWL ontologies. On a more fundamental level, non-standard inferences that support building and maintaining DL knowledge bases are now an important topic of DL research. All these efforts aim at supporting users that are not DL-experts in building and maintaining DL knowledge bases.

In this chapter we have concentrated on very expressive Description Logics that are the formal basis for the web ontology language OWL. For the sake of completeness, we mention here some recent result on inexpressive DLs that are relevant in the context of ontology applications. Several bio-medical ontologies, such as SNOMED [89] and the Gene Ontology [33], are based on rather inexpressive DLs, whose main distinguishing feature is that they disallow value restrictions ($\forall r.C$), but provide for existential restrictions ($\exists r.C$). Recently, it has turned out that such inexpressive DLs with existential restrictions behave much better w.r.t. computational complexity than the corresponding DLs with value restrictions. For example, the subsumption problem in $\mathcal{EL}$, which allows for conjunction, existential restrictions, and the top concept, stays polynomial in the presence of (cyclic or acyclic) concept definitions [2] and even arbitrary GCIs [25]. In [4] it is shown that these polynomiality results also hold for extensions of $\mathcal{EL}$ by constructors that are of interest for ontology applications, such as the bottom concept (which allows disjointness statements to be formulated), nominals, a restricted form of concrete domains, and a restricted form of so-called role-value maps. A first implementation of the polynomial-time subsumption algorithm for such an extension of $\mathcal{EL}$ behaves very well on very large bio-medical ontologies [15].

References

1. Franz Baader. Using automata theory for characterizing the semantics of terminological cycles. *Ann. of Mathematics and Artificial Intelligence*, 18(2–4):175–219, 1996.
2. Franz Baader. Terminological cycles in a description logic with existential restrictions. In Georg Gottlob and Toby Walsh, editors, *Proc. of the 18th Int.*

Joint Conf. on Artificial Intelligence (IJCAI 2003), pages 325–330, Acapulco, Mexico, 2003. Morgan Kaufmann, Los Altos.

3. Franz Baader, Sebastian Brandt, and Ralf Küsters. Matching under side conditions in description logics. In Bernhard Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 213–218, Seattle, Washington, 2001. Morgan Kaufmann.
4. Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, 2005.
5. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
6. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2002. To appear.
7. Franz Baader and Philipp Hanschke. A schema for integrating concrete domains into concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 452–457, 1991.
8. Franz Baader and Philipp Hanschke. Extensions of concept languages for a mechanical engineering application. In *Proc. of the 16th German Workshop on Artificial Intelligence (GWAI'92)*, volume 671 of *Lecture Notes in Computer Science*, pages 132–143. Springer, 1992.
9. Franz Baader and Bernhard Hollunder. A terminological knowledge representation system with complete inference algorithm. In *Proc. of the Workshop on Processing Declarative Knowledge (PDK'91)*, volume 567 of *Lecture Notes in Artificial Intelligence*, pages 67–86. Springer, 1991.
10. Franz Baader and Ralf Küsters. Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions. In *Proc. of the 22nd German Annual Conf. on Artificial Intelligence (KI'98)*, volume 1504 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1998.
11. Franz Baader and Ralf Küsters. Matching in description logics with existential restrictions. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 261–272, 2000.
12. Franz Baader, Ralf Küsters, Alex Borgida, and Deborah L. McGuinness. Matching in description logics. *J. of Logic and Computation*, 9(3):411–447, 1999.
13. Franz Baader, Ralf Küsters, and Ralf Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, pages 96–101, 1999.
14. Franz Baader, Ralf Küsters, and Ralf Molitor. Rewriting concepts using terminologies. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 297–308, 2000.
15. Franz Baader, Carsten Lutz, and Bontawee Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In Ulrich Furbach and Natarajan Shankar, editors, *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 287–291. Springer, 2006.
16. Franz Baader and Ulrike Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69(1):5–40, October 2001.

17. Sean Bechhofer, Ian Horrocks, Carole Goble, and Robert Stevens. OilEd: a reason-able ontology editor for the semantic web. In *Proc. of the 2001 Description Logic Workshop (DL 2001)*, pages 1–9. CEUR (`http://ceur-ws.org/`), 2001.
18. Tim Berners-Lee. *Weaving the Web*. Harpur, San Francisco, 1999.
19. Alexander Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1–2):353–367, 1996.
20. Alexander Borgida and Deborah L. McGuinness. Asking queries about frames. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*, pages 340–349, 1996.
21. Pim Borst, Hans Akkermans, and Jan Top. Engineering ontologies. *International Journal of Human-Computer Studies*, 46:365–406, 1997.
22. Ronald J. Brachman. "Reducing" CLASSIC to practice: Knowledge representation meets reality. In *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'92)*, pages 247–258. Morgan Kaufmann, Los Altos, 1992.
23. Ronald J. Brachman and Hector J. Levesque. The tractability of subsumption in frame-based description languages. In *Proc. of the 4th Nat. Conf. on Artificial Intelligence (AAAI'84)*, pages 34–37, 1984.
24. Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.
25. Sebastian Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In Ramon López de Mántaras and Lorenza Saitta, editors, *Proc. of the 16th Eur. Conf. on Artificial Intelligence (ECAI 2004)*, pages 298–302, 2004.
26. Sebastian Brandt, Ralf Küsters, and Anni-Yasmin Turhan. Approximation and difference in description logics. In D. Fensel, F. Giunchiglia, D. McGuiness, and M.-A. Williams, editors, *Proc. of the 8th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2002)*, pages 203–214. Morgan Kaufmann, Los Altos, 2002.
27. P. Bresciani, E. Franconi, and S. Tessaris. Implementing and testing expressive description logics: Preliminary report. In *Proc. of the 1995 Description Logic Workshop (DL'95)*, pages 131–139, 1995.
28. Martin Buchheit, Francesco M. Donini, Werner Nutt, and Andrea Schaerf. A refined architecture for terminological systems: Terminology = schema + views. *Artificial Intelligence*, 99(2):209–260, 1998.
29. Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.
30. Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Daniele Nardi. Reasoning in expressive description logics. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, chapter 23, pages 1581–1634. Elsevier Science Publishers (North-Holland), Amsterdam, 2001.
31. Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Description logic framework for information integration. In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 2–13, 1998.
32. William W. Cohen, Alex Borgida, and Haym Hirsh. Computing least common subsumers in description logics. In William Swartout, editor, *Proc. of*

the 10th Nat. Conf. on Artificial Intelligence (AAAI'92), pages 754–760. AAAI Press/The MIT Press, 1992.
33. The Gene Ontology Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.
34. Giuseppe De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms*. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1995.
35. Giuseppe De Giacomo and Maurizio Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI'94)*, pages 205–212, 1994.
36. Giuseppe De Giacomo and Maurizio Lenzerini. Concept language with number restrictions and fixpoints, and its relationship with μ-calculus. In *Proc. of the 11th Eur. Conf. on Artificial Intelligence (ECAI'94)*, pages 411–415, 1994.
37. Giuseppe De Giacomo and Maurizio Lenzerini. TBox and ABox reasoning in expressive description logics. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*, pages 316–327, 1996.
38. Francesco M. Donini, Bernhard Hollunder, Maurizio Lenzerini, Alberto Marchetti Spaccamela, Daniele Nardi, and Werner Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 2–3:309–327, 1992.
39. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The complexity of concept languages. In *Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'91)*, pages 151–162, 1991.
40. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. Tractable concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 458–463, 1991.
41. Jon Doyle and Ramesh S. Patil. Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence*, 48:261–297, 1991.
42. D. Fensel, F. van Harmelen, M. Klein, H. Akkermans, J. Broekstra, C. Fluit, J. van der Meer, H.-P. Schnurr, R. Studer, J. Hughes, U. Krohn, J. Davies, R. Engels, B. Bremdal, F. Ygge, T. Lau, B. Novotny, U. Reimer, and I. Horrocks. On-To-Knowledge: Ontology-based tools for knowledge management. In *Proceedings of the eBusiness and eWork 2000 (eBeW'00 Conference*, October 2000.
43. Erich Grädel. Guarded fragments of first-order logic: A perspective for new description logics? In *Proc. of the 1998 Description Logic Workshop (DL'98)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-11/`, 1998.
44. Erich Grädel. On the restraining power of guards. *J. of Symbolic Logic*, 64:1719–1742, 1999.
45. Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.
46. Erich Grädel, Martin Otto, and Eric Rosen. Two-variable logic with counting is decidable. In *Proc. of the 12th IEEE Symp. on Logic in Computer Science (LICS'97)*, 1997.
47. Thomas Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

48. Volker Haarslev and Ralf Möller. RACE system description. In *Proc. of the 1999 Description Logic Workshop (DL'99)*, pages 130–132. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-22/`, 1999.
49. Volker Haarslev and Ralf Möller. Expressive ABox reasoning with number restrictions, role hierarchies, and transitively closed roles. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 273–284, 2000.
50. Volker Haarslev and Ralf Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 701–705. Springer, 2001.
51. Patrick Hayes. RDF model theory. W3C Working Draft, April 2002. `http://www.w3.org/TR/rdf-mt/`.
52. Bernhard Hollunder, Werner Nutt, and Manfred Schmidt-Schauß. Subsumption algorithms for concept description languages. In *Proc. of the 9th Eur. Conf. on Artificial Intelligence (ECAI'90)*, pages 348–353, London (United Kingdom), 1990. Pitman.
53. Ian Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.
54. Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 57–67. AAAI Press, 2006.
55. Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.
56. Ian Horrocks and Ulrike Sattler. A description logic with transitive and inverse roles and role hierarchies. *J. of Logic and Computation*, 9(3):385–410, 1999.
57. Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 448–453, 2005.
58. Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In Harald Ganzinger, David McAllester, and Andrei Voronkov, editors, *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer, 1999.
59. Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for very expressive description logics. *J. of the Interest Group in Pure and Applied Logic*, 8(3):239–264, 2000.
60. Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In David McAllester, editor, *Proc. of the 17th Int. Conf. on Automated Deduction (CADE 2000)*, volume 1831 of *Lecture Notes in Computer Science*, pages 482–496. Springer, 2000.
61. Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca-Grau, and James Hendler. SWOOP: a web ontology editing browser. *J. of Web Semantics*, 4(2), 2005.
62. Holger Knublauch, Ray Fergerson, Natalya Noy, and Mark Musen. The Protégé OWL Plugin: An open development environment for semantic web applications. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *Proc. of the 2004 International Semantic Web Conference (ISWC 2004)*, number 3298 in Lecture Notes in Computer Science, pages 229–243. Springer, 2004.

63. Ralf Küsters and Alex Borgida. What's in an Attribute? Consequences for the Least Common Subsumer. *J. of Artificial Intelligence Research*, 14:167–203, 2001.
64. Ralf Küsters and Ralf Molitor. Approximating Most Specific Concepts in Description Logics with Existential Restrictions. In F. Baader, editor, *Proc. of the Joint German/Austrian Conference on Artificial Intelligence, 24th German / 9th Austrian Conference on Artificial Intelligence (KI 2001)*, volume 2174 of *Lecture Notes in Artificial Intelligence*. Springer, 2001.
65. Ralf Küsters and Ralf Molitor. Computing Least Common Subsumers in ALEN. In Bernard Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 219–224. Morgan Kaufmann, Los Altos, 2001.
66. Carsten Lutz. NEXPTIME-complete description logics with concrete domains. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 45–60. Springer, 2001.
67. Carsten Lutz. Description logics with concrete domains—a survey. In *Advances in Modal Logics Volume 4*. World Scientific Publishing Co. Pte. Ltd., 2003.
68. Robert MacGregor. The evolving technology of classification-based knowledge representation systems. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 385–400. Morgan Kaufmann, Los Altos, 1991.
69. Eric Mays, Robert Dionne, and Robert Weida. K-Rep system overview. *SIGART Bull.*, 2(3):93–97, 1991.
70. Ralf Molitor. *Unterstützung der Modellierung verfahrenstechnischer Prozesse durch Nicht-Standardinferenzen in Beschreibungslogiken*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2000. In German.
71. Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Artificial Intelligence*. Springer, 1990.
72. Bernhard Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.
73. Bernhard Nebel. Terminological cycles: Semantics and computational properties. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 331–361. Morgan Kaufmann, Los Altos, 1991.
74. Leszek Pacholski, Wieslaw Szwast, and Lidia Tendera. Complexity of two-variable logic with counting. In *Proc. of the 12th IEEE Symp. on Logic in Computer Science (LICS'97)*, pages 318–327. IEEE Computer Society Press, 1997.
75. Jeff Z. Pan and Ian Horrocks. Semantic web ontology reasoning in the $\mathcal{SHOQ}(\mathbf{D_n})$ description logic. In *Proc. of the 2002 Description Logic Workshop (DL 2002)*, 2002.
76. Peter F. Patel-Schneider. DLP. In *Proc. of the 1999 Description Logic Workshop (DL'99)*, pages 9–13. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-22/`, 1999.
77. Peter F. Patel-Schneider, Deborah L. McGuiness, Ronald J. Brachman, Lori Alperin Resnick, and Alexander Borgida. The CLASSIC knowledge representation system: Guiding principles and implementation rational. *SIGART Bull.*, 2(3):108–113, 1991.
78. Christof Peltason. The BACK system — an overview. *SIGART Bull.*, 2(3):114–119, 1991.
79. A. Rector, S. Bechhofer, C. A. Goble, I. Horrocks, W. A. Nowlan, and W. D. Solomon. The GRAIL concept modelling language for medical terminology. *Artificial Intelligence in Medicine*, 9:139–171, 1997.

80. Ulrike Sattler. *Terminological knowledge representation systems in a process engineering application*. PhD thesis, RWTH Aachen, 1998.
81. Ulrike Sattler. Description logics for the representation of aggregated objects. In *Proc. of the 14th Eur. Conf. on Artificial Intelligence (ECAI 2000)*, 2000.
82. Andrea Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13(2):141–176, 1994.
83. Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 466–471, 1991.
84. Klaus Schild. *Querying Knowledge and Data Bases by a Universal Description Logic with Recursion*. PhD thesis, Universität des Saarlandes, Germany, 1995.
85. Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
86. Stefan Schulz and Udo Hahn. Parts, locations, and holes - formal reasoning about anatomical structures. In *Proc. of AIME 2001*, volume 2101 of *Lecture Notes in Artificial Intelligence*. Springer, 2001.
87. Evren Sirin, Bernardo Cuenca Grau, and Bijan Parsia. From wine to water: Optimizing description logic reasoning for nominals. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 90–99. AAAI Press, 2006.
88. Evren Sirin and Bijan Parsia. Pellet: An OWL DL reasoner. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*, 2004.
89. K.A. Spackman, K.E. Campbell, and R.A. Cote. SNOMED RT: A reference terminology for health care. *J. of the American Medical Informatics Association*, pages 640–644, 1997. Fall Symposium Supplement.
90. Robert Stevens, Ian Horrocks, Carole Goble, and Sean Bechhofer. Building a reason-able bioinformatics ontology using OIL. In *Proceedings of the IJCAI-2001 Workshop on Ontologies and Information Sharing*, pages 81–90. CEUR (`http://ceur-ws.org/`), 2001.
91. Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, 2001.
92. Dmitry Tsarkov and Ian Horrocks. FaCT++ description logic reasoner: System description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer, 2006.

Index

Author Index